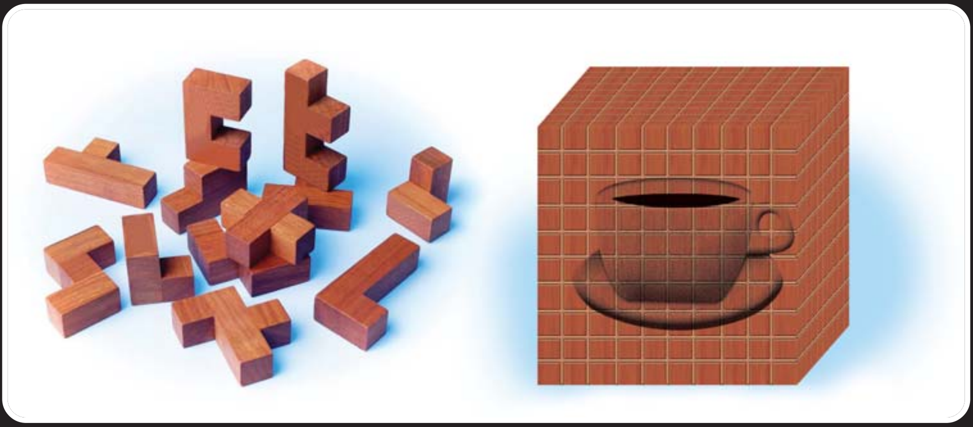


If you want to keep your code clean and concise to make it easier to understand, modify and extend, then go for refactoring.

Java Refactoring Part–I With Eclipse

Refactoring is the process of changing the internal structure of a program (like removing redundancies, eliminating unused functions, rejuvenating obsolete designs, etc) without changing its external behaviour. There are a number of good tools with which you can do Java refactoring—Eclipse, JRefactory and JEdit, to name a few.

In this two-part article, we will explore how you could refactor Java code automatically, using one of the most commonly used tools—Eclipse. Eclipse is a popular open source extensible IDE for Java and other languages. It provides preview and instant validation, in addition to an 'undo/redo' facility for refactoring. You can get the latest version of Eclipse from [http://www.eclipse.org/downloads/index.php]

Figure 1 shows the Refactor dropdown menu in Eclipse. You can see that it has four sections. The first section has the 'Undo' and 'Redo' functions on it, and the other three sections relate to the different types of refactoring operations that you can do with Eclipse.

Java Refactoring with Eclipse can be best explained by applying various refactoring techniques to a sample program shown in Figure 1. The idea is to make the code cleaner and more understandable.

In this article, we will discuss a few refactoring operations (like encapsulate field, change method signature and extract method). The first refactoring technique that we will apply to our sample program is 'Encapsulate Field'. The idea is to add 'setter and getter' methods for the class field members. To encapsulate a field, select the field name and click *Encapsulate Field* menu item under *Refactor* menu. Enter 'setter' and 'getter' method names and modifiers in the pop-up dialog as shown in Figure 2. Figure 3 shows the preview dialog. Click on the OK button to accept the changes. Similarly, define the 'getter and setter' methods for other fields like smtpPort, toAddr, fromAddr, subject and body.

Since we have defined the 'getter and setter' methods for class field members, we will modify the constructor so that it becomes a 'no argument' constructor. To change a method signature, select the method name (in our case, it is the constructor) and click *Change Method Signature* under the *Refactor* menu. Delete all the method parameters as shown in Figure 4. Figures 5 and 6 show the preview dialog.

The next refactoring technique that we will apply to our sample program is the 'Extract Method'. We will use this technique to extract two methods—one that connects to the SMTP server and another method that disconnects from the SMTP. To extract a method, select the section of code and click on the *Extract Method* menu item under the *Refactor* menu. Figures 7 and 8 show the preview dialog. They also show that Eclipse has intelligently extracted the method and updated all the references.

Similarly, extract the method disconnectSMTP as shown in Figures 9 and 10. A sample SMTP e-mail client program in Java has been included in the LFY CD.

In the first part of this article we have seen how to apply the various refactoring techniques from Eclipse to make the code cleaner and easier to maintain. In the second part, we will take a look at the rest of the refactoring operations. Until then, happy refactoring!

References

http://www.refactoring.com/
www.cs.umanitoba.ca/~eclipse/13-Refactoring.pdf
http://c2.com/cgi/wiki?WhatIsRefactoring

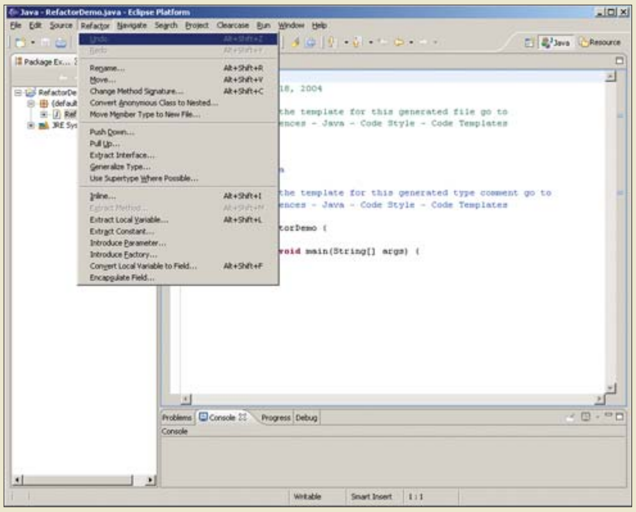
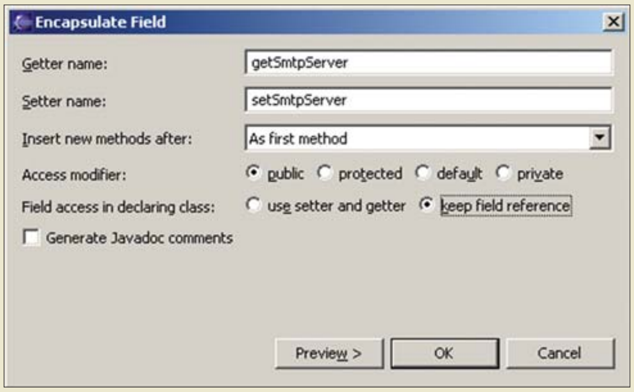
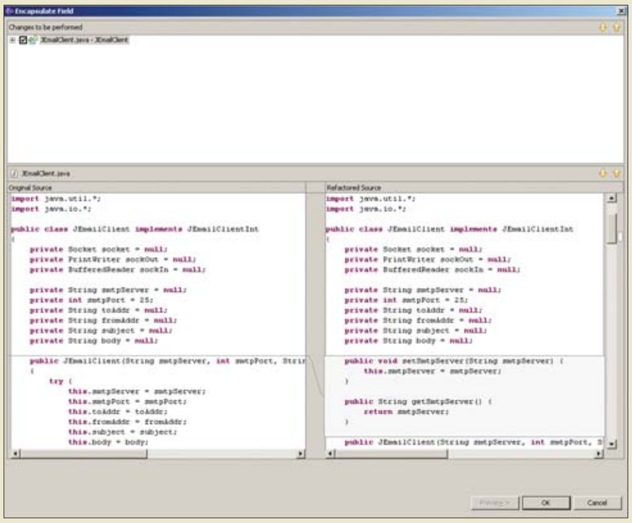

Java - RefactorDemo.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Clearcase Run Window Help
Undo Alt+Shift+Z
Redo Alt+Shift+Y
Rename... Alt+Shift+R
Move... Alt+Shift+V
Change Method Signature... Alt+Shift+C
Convert Anonymous Class to Nested...
Move Member Type to New File...
Push Down...
Pull Up...
Extract Interface...
Generalize Type...
Use Supertype Where Possible...
Inline... Alt+Shift+I
Extract Method... Alt+Shift+M
Extract Local Variable... Alt+Shift+L
Extract Constant...
Introduce Parameter...
Introduce Factory...
Convert Local Variable to Field... Alt+Shift+F
Encapsulate Field...
Package Ex...
Problems Console Progress Debug
Writable Smart Insert 1 : 1

Figure 1: Refactoring with Eclipse


Encapsulate Field

Getter name:	getSmtpServer
Setter name:	setSmtpServer
Insert new methods after:	As first method
Access modifier:	[✓ public] protected default private
Field access in declaring class:	use setter and getter [✓ keep field reference]
Generate Javadoc comments	☐

Figure 2: Encapsulate Field pop-up dialog



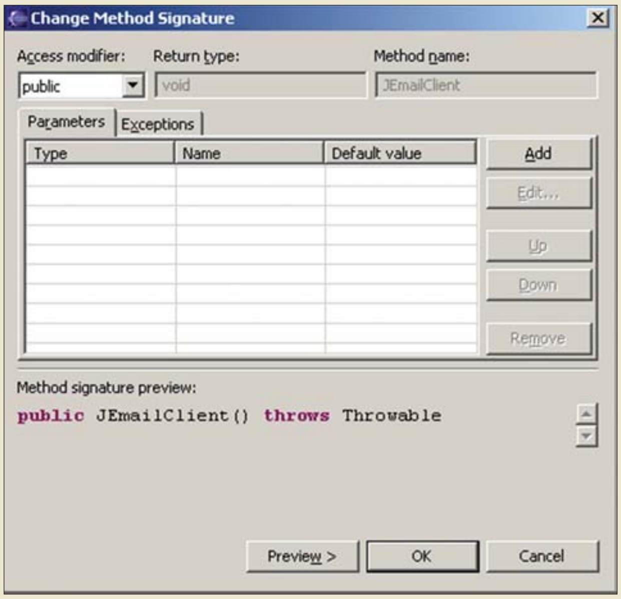
Original Source:

```
import java.util.*;
import java.io.*;

public class JEmailClient implements JEmailClientInt
{
    private Socket socket = null;
    private PrintWriter sockOut = null;
    private BufferedReader sockIn = null;

    private String smtpServer = null;
    private int smtpPort = 25;
    private String toAddr = null;
    private String fromAddr = null;
    private String subject = null;
    private String body = null;

    public JEmailClient(String smtpServer, int smtpPort, Strin
    {
        try {
            this.smtpServer = smtpServer;
            this.smtpPort = smtpPort;
            this.toAddr = toAddr;
            this.fromAddr = fromAddr;
            this.subject = subject;
            this.body = body;
```

Refactored Source:

```
import java.util.*;
import java.io.*;

public class JEmailClient implements JEmailClientInt
{
    private Socket socket = null;
    private PrintWriter sockOut = null;
    private BufferedReader sockIn = null;

    private String smtpServer = null;
    private int smtpPort = 25;
    private String toAddr = null;
    private String fromAddr = null;
    private String subject = null;
    private String body = null;

    public void setSmtpServer(String smtpServer) {
        this.smtpServer = smtpServer;
    }

    public String getSmtpServer() {
        return smtpServer;
    }

    public JEmailClient(String smtpServer, int smtpPort, S
```

Figure 3: Encapsulate field preview dialog

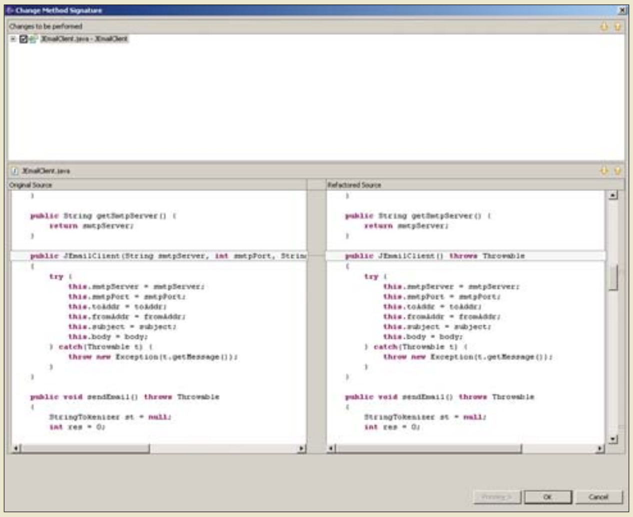

Change Method Signature

Access modifier:	Return type:	Method name:
public	void	JEmailClient

Method signature preview:

```
public JEmailClient() throws Throwable
```

Figure 4: Change method signature pop-up dialog



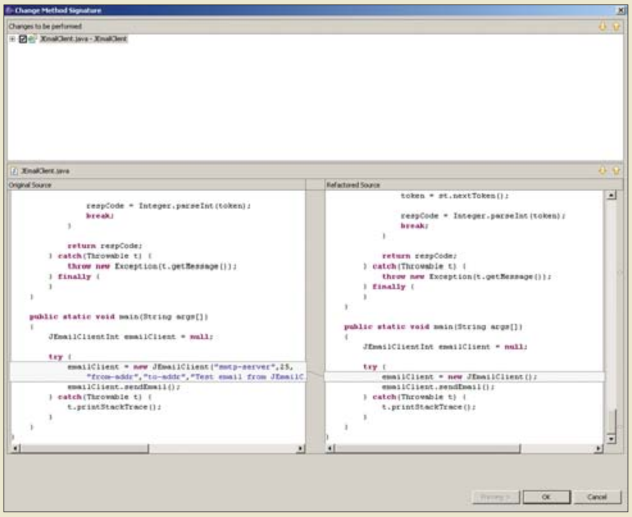
Original Source:

```
    }

    public String getSmtpServer() {
        return smtpServer;
    }

    public JEmailClient(String smtpServer, int smtpPort, Strin
    {
        try {
            this.smtpServer = smtpServer;
            this.smtpPort = smtpPort;
            this.toAddr = toAddr;
            this.fromAddr = fromAddr;
            this.subject = subject;
            this.body = body;
        } catch(Throwable t) {
            throw new Exception(t.getMessage());
        }
    }

    public void sendEmail() throws Throwable
    {
        StringTokenizer st = null;
        int res = 0;
```

Refactored Source:

```
    }

    public String getSmtpServer() {
        return smtpServer;
    }

    public JEmailClient() throws Throwable
    {
        try {
            this.smtpServer = smtpServer;
            this.smtpPort = smtpPort;
            this.toAddr = toAddr;
            this.fromAddr = fromAddr;
            this.subject = subject;
            this.body = body;
        } catch(Throwable t) {
            throw new Exception(t.getMessage());
        }
    }

    public void sendEmail() throws Throwable
    {
        StringTokenizer st = null;
        int res = 0;
```

Figure 5: Change method signature preview dialog



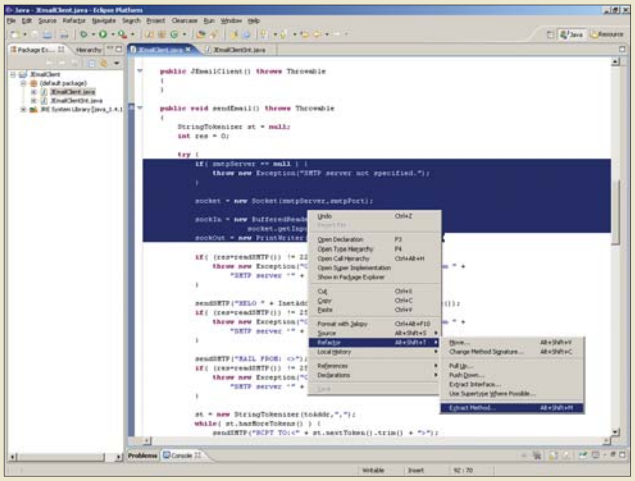
Original Source:

```
                respCode = Integer.parseInt(token);
                break;
            }

            return respCode;
        } catch(Throwable t) {
            throw new Exception(t.getMessage());
        } finally {
        }
    }

    public static void main(String args[])
    {
        JEmailClientInt emailClient = null;

        try {
            emailClient = new JEmailClient("smtp-server",25,
                "from-addr","to-addr","Test email from JEmailC
            emailClient.sendEmail();
        } catch(Throwable t) {
            t.printStackTrace();
        }
    }
}
```

Refactored Source:

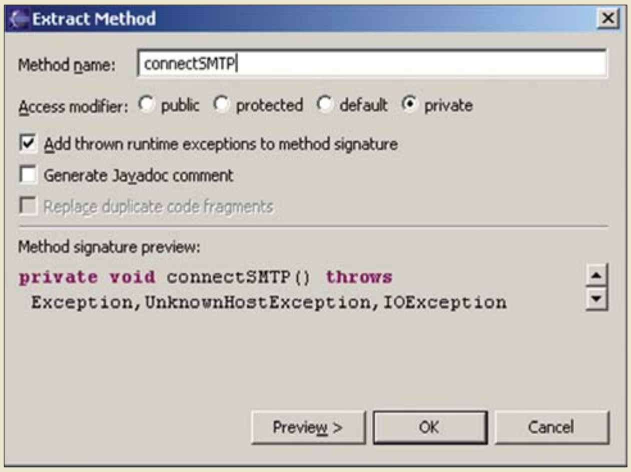
```
                token = st.nextToken();

                respCode = Integer.parseInt(token);
                break;
            }

            return respCode;
        } catch(Throwable t) {
            throw new Exception(t.getMessage());
        } finally {
        }
    }

    public static void main(String args[])
    {
        JEmailClientInt emailClient = null;

        try {
            emailClient = new JEmailClient();
            emailClient.sendEmail();
        } catch(Throwable t) {
            t.printStackTrace();
        }
    }
}
```

Figure 6: Change method signature preview dialog



```
public JEmailClient() throws Throwable
{
}

public void sendEmail() throws Throwable
{
    StringTokenizer st = null;
    int res = 0;

    try {
        if( smtpServer == null ) {
            throw new Exception("SMTP server not specified.");
        }

        socket = new Socket(smtpServer,smtpPort);

        sockIn = new BufferedReade
                socket.getInpu
        sockOut = new PrintWriter(

        if( (res=readSMTP()) != 22
            throw new Exception("C                              " +
                "SMTP server '" +
        }

        sendSMTP("HELO " + InetAdd                              ());
        if( (res=readSMTP()) != 25
            throw new Exception("C                              " +
                "SMTP server '" +
        }

        sendSMTP("MAIL FROM: <>");
        if( (res=readSMTP()) != 25
            throw new Exception("C
                "SMTP server '" +
        }

        st = new StringTokenizer(toAddr,",");
        while( st.hasMoreTokens() ) {
            sendSMTP("RCPT TO:<" + st.nextToken().trim() + ">");
```

Figure 7: Extract method refactoring operation



Method name: connectSMTP

Access modifier: ( ) public ( ) protected ( ) default [✓ private]

[✓] Add thrown runtime exceptions to method signature

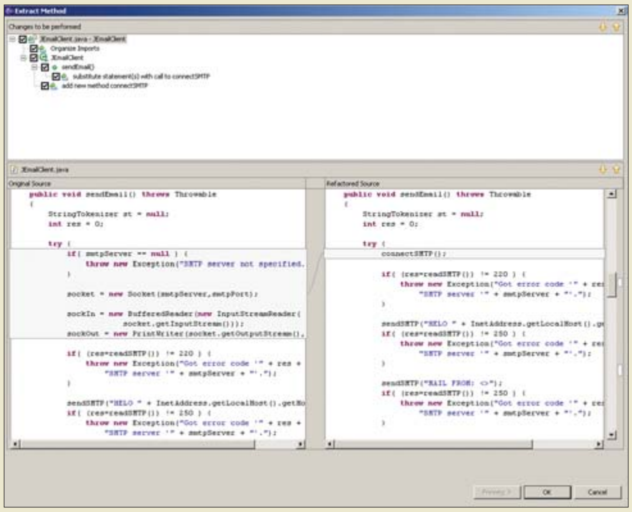
[ ] Generate Javadoc comment

[ ] Replace duplicate code fragments

Method signature preview:

```
private void connectSMTP() throws
 Exception,UnknownHostException,IOException
```

Figure 8: Extract method pop-up dialog



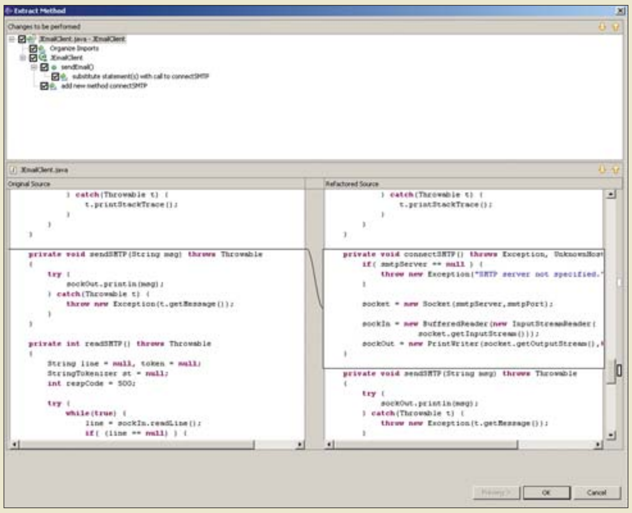
Original Source:

```
public void sendEmail() throws Throwable
{
    StringTokenizer st = null;
    int res = 0;

    try {
        if( smtpServer == null ) {
            throw new Exception("SMTP server not specified.
        }

        socket = new Socket(smtpServer,smtpPort);

        sockIn = new BufferedReader(new InputStreamReader(
                socket.getInputStream()));
        sockOut = new PrintWriter(socket.getOutputStream(),

        if( (res=readSMTP()) != 220 ) {
            throw new Exception("Got error code '" + res +
                "SMTP server '" + smtpServer + "'.");
        }

        sendSMTP("HELO " + InetAddress.getLocalHost().getHo
        if( (res=readSMTP()) != 250 ) {
            throw new Exception("Got error code '" + res +
                "SMTP server '" + smtpServer + "'.");
```

Refactored Source:

```
public void sendEmail() throws Throwable
{
    StringTokenizer st = null;
    int res = 0;

    try {
        connectSMTP();

        if( (res=readSMTP()) != 220 ) {
            throw new Exception("Got error code '" + res
                "SMTP server '" + smtpServer + "'.");
        }

        sendSMTP("HELO " + InetAddress.getLocalHost().ge
        if( (res=readSMTP()) != 250 ) {
            throw new Exception("Got error code '" + res
                "SMTP server '" + smtpServer + "'.");
        }

        sendSMTP("MAIL FROM: <>");
        if( (res=readSMTP()) != 250 ) {
            throw new Exception("Got error code '" + res
                "SMTP server '" + smtpServer + "'.");
        }
```

Figure 9: Extract method preview dialog



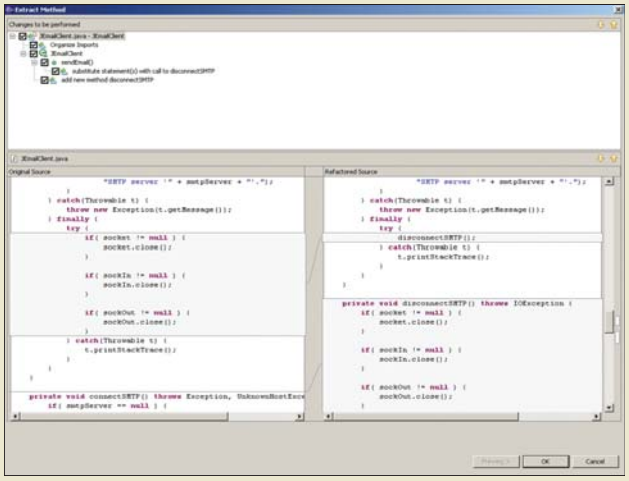
Original Source:

```
            } catch(Throwable t) {
                t.printStackTrace();
            }
        }
    }

    private void sendSMTP(String msg) throws Throwable
    {
        try {
            sockOut.println(msg);
        } catch(Throwable t) {
            throw new Exception(t.getMessage());
        }
    }

    private int readSMTP() throws Throwable
    {
        String line = null, token = null;
        StringTokenizer st = null;
        int respCode = 500;

        try {
            while(true) {
                line = sockIn.readLine();
                if( (line == null) ) {
```

Refactored Source:

```
            } catch(Throwable t) {
                t.printStackTrace();
            }
        }
    }

    private void connectSMTP() throws Exception, UnknownHost
        if( smtpServer == null ) {
            throw new Exception("SMTP server not specified."
        }

        socket = new Socket(smtpServer,smtpPort);

        sockIn = new BufferedReader(new InputStreamReader(
                socket.getInputStream()));
        sockOut = new PrintWriter(socket.getOutputStream(),
    }
    private void sendSMTP(String msg) throws Throwable
    {
        try {
            sockOut.println(msg);
        } catch(Throwable t) {
            throw new Exception(t.getMessage());
        }
```

Figure 10: Extract method preview dialog



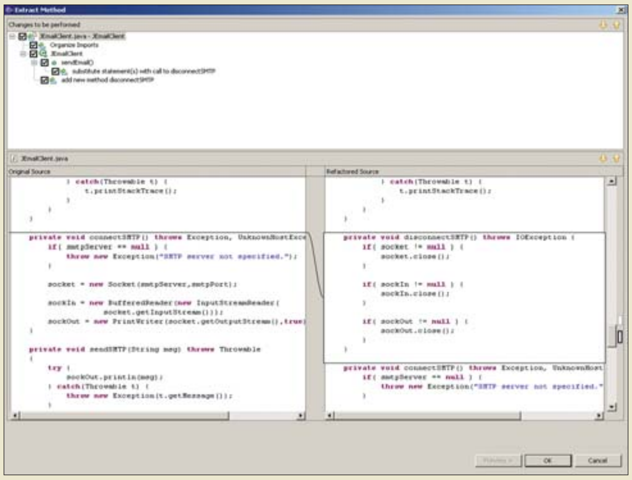
Original Source:

```
                "SMTP server '" + smtpServer + "'.");
            }
        } catch(Throwable t) {
            throw new Exception(t.getMessage());
        } finally {
            try {
                if( socket != null ) {
                    socket.close();
                }

                if( sockIn != null ) {
                    sockIn.close();
                }

                if( sockOut != null ) {
                    sockOut.close();
                }
            } catch(Throwable t) {
                t.printStackTrace();
            }
        }
    }
    private void connectSMTP() throws Exception, UnknownHostExce
        if( smtpServer == null ) {
```

Refactored Source:

```
                "SMTP server '" + smtpServer + "'.");
            }
        } catch(Throwable t) {
            throw new Exception(t.getMessage());
        } finally {
            try {
                disconnectSMTP();
            } catch(Throwable t) {
                t.printStackTrace();
            }
        }
    }

    private void disconnectSMTP() throws IOException {
        if( socket != null ) {
            socket.close();
        }

        if( sockIn != null ) {
            sockIn.close();
        }

        if( sockOut != null ) {
            sockOut.close();
        }
```

Figure 11: Extract method preview dialog



Original Source:

```
            } catch(Throwable t) {
                t.printStackTrace();
            }
        }
    }
    private void connectSMTP() throws Exception, UnknownHostExce
        if( smtpServer == null ) {
            throw new Exception("SMTP server not specified.");
        }

        socket = new Socket(smtpServer,smtpPort);

        sockIn = new BufferedReader(new InputStreamReader(
                socket.getInputStream()));
        sockOut = new PrintWriter(socket.getOutputStream(),true)
    }

    private void sendSMTP(String msg) throws Throwable
    {
        try {
            sockOut.println(msg);
        } catch(Throwable t) {
            throw new Exception(t.getMessage());
        }
```

Refactored Source:

```
            } catch(Throwable t) {
                t.printStackTrace();
            }
        }
    }
    private void disconnectSMTP() throws IOException {
        if( socket != null ) {
            socket.close();
        }

        if( sockIn != null ) {
            sockIn.close();
        }

        if( sockOut != null ) {
            sockOut.close();
        }
    }
    private void connectSMTP() throws Exception, UnknownHost
        if( smtpServer == null ) {
            throw new Exception("SMTP server not specified."
        }
```

Figure 12: Extract method preview dialog

LFY

(to be continued...)

By: Raja R K. The author is a lead engineer working with HCL Technologies (Cisco Systems Offshore Development Centre) in Chennai. He can be contacted at: rajark_hcl@yahoo.co.in